Parallelization of FEMWATER
An Unstructured Finite Element Groundwater Model

F.T. Tracy* and J.P. Holland
US Army Engineer Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180

I.L. Carpenter and R.W. Numrich
Silicon Graphics / Cray Research
655E Lone Oak Dr.
Eagan, MN 55121

Abstract

When starting from scratch, it is challenging to obtain good scalability in unstructured finite element (FE) models. It is even more difficult to parallelize older existing codes such as FEMWATER. If done the wrong way, the results of scaled speed-up tests can be very discouraging. There are several factors that can cause poor results. These include grid partitioning, the preparation program, boundary conditions, sparse matrix storage format, ghost nodes, border elements, updating, solvers for linear and nonlinear equations, and data types. All these issues were addressed in the parallelization of FEMWATER. When going from 4 to 8 PE's, with the grid size going from 62,500 nodes to 125,000 nodes, a scaled speed-up of 95.8% was achieved on the CEWES Cray T3E for a steady-state problem. Scaling both the grid to 1,000,000 nodes and PE's to 64 also gave good results. A time-dependent, unsaturated flow problem was also successfully run using 198 PE's. The same code that runs on the Cray T3E also runs without change on the IBM SP system. The purpose of this paper is to describe the techniques used to achieve these results by addressing the items mentioned above and in the list. Note that only subsurface flow is discussed in this paper. Each item will now be briefly described.

1. *Grid Partitioning.* Making a balanced load by using a good grid partitioning is essential to achieving efficient scaling.

2. *Preparation.* The program to convert the global data into data for each processing element (PE) is very important and with care it can be reduced to very short running times.

3. *Boundary Conditions, Sparse Matrix Storage Forma, Ghost Nodes, and Border Elements.* Many FE programs apply boundary conditions at the element level, so they do not store a global stiffness matrix. However, FEMWATER creates a global stiffness matrix in sparse format and modifies it when a specified pressure boundary condition is given. A boundary condition applied to a node in one PE can generate changes for the stiffness matrix stored in neighboring PE's. This can lead to significant communication times. Deciding how to handle the elements that cross PE boundaries is of critical importance to getting good performance.

4. *Updating*. Updating the ghost nodes using MPI can give inconsistent results on different machines. Sometimes what works on one system completely fails on another. What is presented here is what worked well across the different platforms.

5. *Solvers*. Solvers for linear and nonlinear equations are a significant research topic in and of themselves. What will be presented here are results for a relaxation solver used on a steady-state and difficult unsaturated flow problem.

6. *Data Types.* FEMWATER runs in double precision mode on a workstation. Getting the MPI to work correctly on multiple platforms requires the use of data types that will be described.

1. Introduction

1.1 Description of FEMWATER

FEMWATER (Lin, Richards, Talbot, Yeh, Cheng, Cheng, and Jones, 1997) is a legacy unstructured grid, finite element code that models groundwater flow and transport. The flow can be unsaturated, so added to the geometric complexity is a system of nonlinear equations to solve at each time step. Data files for FEMWATER are typically generated from the DoD Groundwater Modeling System (GMS) (Groundwater Modeling Team, 1998), and they include a geometry file, a boundary condition and parameter file, and a super file containing file names.

1.2 Challenges

When starting from scratch, it is challenging to obtain good scalability in unstructured finite element models. It is even more difficult to parallelize older existing codes such as FEMWATER. If done the wrong way, the results of scaled speed-up tests can be very discouraging. There are several factors that can cause poor results. These include grid partitioning, the preparation program, boundary conditions, sparse matrix storage format, ghost nodes, border elements, updating, solvers for linear and nonlinear equations, and data types. All these issues were addressed in the parallelization of FEMWATER with excellent results, and these items will now be discussed in greater detail.

2. Parallelization of FEMWATER

The techniques used to successfully parallelize FEMWATER are now described in more detail. Only flow modeling will be discussed in this paper.

2.1 Grid Partitioning

Partitioning the grid so each PE has roughly the same load is critical for good performance. METIS (Karypis, 1998) was chosen as it takes only a few seconds to run and is readily available. Table 1 shows results for the steady-state problem for 4, 8, 16, and 32 PE's where a block partition was compared to a METIS partition for 4, 8, and 16 PE's. The block partition is excellent, but METIS consistently out-performed it.

2.2 Preparation

The basic paradigm is to change the original FEMWATER code as little as possible. Therefore, a preparation code must be written to start with the original input files and create input files for each PE that look the same as the original files but have only the data needed by each respective PE. This preparation program can take lots of time to run on a singe PE, and it can also get involved with lengthy sorting. However, when run in parallel and with care taken to keep track of local, ghost, and global node and element numbers, the preparation code can run very efficiently. Table 2 gives times for 4, 8, and 64 PE's. Less than one minute for 1,000,000 nodes is excellent as unsaturated flow problems can take hours.

The preparation process consists of: (1) determining the local, global, and ghost node and element numbers for each PE and writing each PE a file: (2) writing a geometry file for each PE that looks like the original: and (3) processing and writing a parameter and boundary condition file for each PE that looks like the original.

2.3 Boundary Conditions, Sparse Matrix Storage Format, Ghost Nodes, and Border Elements

Many recently-developed FE programs apply boundary conditions at the element level, so they do not store a global stiffness matrix. However, FEMWATER creates a global symmetric stiffness matrix in sparse format and modifies it when a specified pressure head boundary condition is given to maintain that symmetry. Thus, a boundary condition applied to a node in one PE can generate changes for the stiffness matrix stored in neighboring PE’s. Figure 1 shows what happens to the **[K]{h}** = **{Q}** system of equations when a six-node problem is partitioned into two parts for two PE’s when having pressure head, $h = h_0$, specified at node 3. The terms in italics are not actually stored. Row 3, column 3, and the right-hand side are modified. This can lead to significant communication times as it can cross several PE boundaries. The solution that leads to ***no*** addition of parallel code in the process of applying boundary conditions is correctly handling the ghost nodes and border elements where nodes of that element belong to different PE’s (see Figure 2). First, the system of equations shown in Figure 1 is assembled one element at a time, so assemble the border elements in both PE’s. Next, the preparation program outputs a file identical to FEMWATER format for the parameter and boundary condition data. In this process keep boundary condition data for both the owned nodes and ghost nodes. For the nodes owned by the respective PE, ***all*** boundary conditions will then be done correctly. This is because the stiffness matrix only contains non-zero terms for neighboring nodes. One would first think that doing the element twice is very inefficient, but it is actually “free” because, as seen in Figure 2, rather than PE 0 doing six elements and PE 1 doing three, both PE 0 and PE 1 assemble six elements, giving a perfect load balance. FEMWATER now works without barriers, and the solver only works on the owned nodes by the respective PE’s before updating the shared data.

Table 1. METIS versus Block Partition for the T3E

	Grid Size	**No. of PE’s**	**Solver Iterations**	**Time (sec)**	**Scaled Speed-Up (%)**
Block Partition	50 X 50 X 25	4	1148	134.0	~
METIS	50 X 50 X 25	4	1150	133.6	~
Block Partition	50 X 50 X 50	8	1160	147.1	91.1
METIS	50 X 50 X 50	8	1163	139.5	95.8
Block Partition	50 X 50 X 100	16	1166	145.1	92.4
METIS	50 X 50 X 100	16	1168	144.2	92.6
Block Partition	50 X 100 X 100	32	1173	149.5	89.6

Table 2. Preparation Program Times

Grid Size	**No. of PE’s**	**Time (sec)**
50 X 50 X 25	4	4.3
50 X 50 X 50	8	7.0
100 X 100 X 100	64	39.6

$$
\begin{array}{c}
\text{PE 0} \\
\text{? ? ?} \\
\text{PE 1}
\end{array}
\begin{bmatrix}
k_{11} & k_{12} & 0 & k_{14} & k_{15} & k_{16} \\
\mathit{k_{12}} & k_{22} & 0 & k_{24} & k_{25} & k_{26} \\
\mathit{0} & \mathit{0} & 1 & 0 & 0 & 0 \\
\mathit{k_{14}} & \mathit{k_{24}} & \mathit{0} & k_{44} & k_{45} & k_{46} \\
\mathit{k_{15}} & \mathit{k_{25}} & \mathit{0} & \mathit{k_{45}} & k_{55} & k_{56} \\
\mathit{k_{16}} & \mathit{k_{26}} & \mathit{0} & \mathit{k_{46}} & \mathit{k_{56}} & k_{66}
\end{bmatrix}
\begin{Bmatrix}
h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6
\end{Bmatrix}
=
\begin{Bmatrix}
Q_1 - k_{13}h_0 \\ Q_2 - k_{23}h_0 \\ h_0 \\ Q_4 - k_{34}h_0 \\ Q_5 - k_{35}h_0 \\ Q_6 - k_{36}h_0
\end{Bmatrix}
\begin{array}{c}
\text{PE 0} \\
\text{? ? ?} \\
\text{PE 1}
\end{array}
$$

Figure 1. Specified Head Boundary Condition

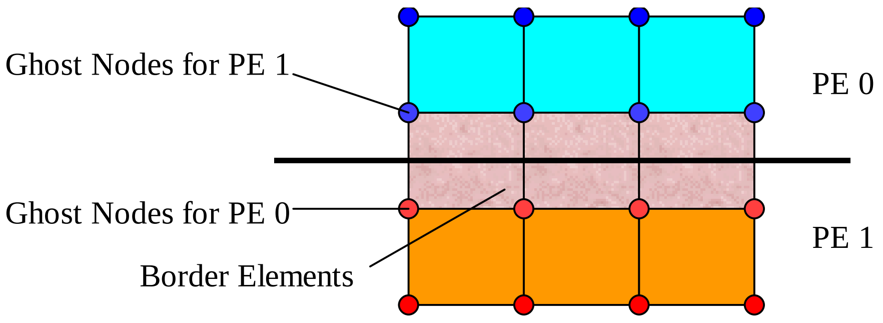

Figure 2. Ghost Nodes and Border Elements

2.4 Updating

MPI behaves remarkably differently on different systems. After trying several different ways to update information for the ghost nodes from other PE's, the code in Ta ble 3 was chosen as it works well for both the T3E and SP. Sorting the ghost nodes in each PE allows ghost data to be received in consecutive order.

2.5 Solvers

The solver is critical to the successful parallelization of any implicit numerical code. T able 1 shows good results for the steady-state problem. The solver used here is a modified alternating direction relaxation method. Note that the scaled speed-up results are quite good. However, for 32 PE's or less, the grid was only increased in a direction different from the flow direction. Had the grid been increased in the direct ion of flow (+x), the number of iterations would have increased somewhat. Experience with a multi -grid pre-conditioner to a conjugate gradient solver on a structured grid has shown that this increase in iterations was almost nonexistent there (from 17 to 19).

Table 3. MPI Code

```
      Do for each sending_PE to myid
C
c         Receive num values in the vector v starting at nst from
c         sending_PE.
C
        Call MPI_IRECV (v(nst), num, MPI_REAL8, sending_PE, itag,
c         MPI_COMM_WORLD, ireq(i), ierror)
      End do
C
      Do for each receiving_PE from myid
C
c         After the data has been packed in buff, send num values to
c         receiving_PE.
C
          Call MPI_SEND (buff, num, MPI_REAL8, receiving_PE, itag,
     &      MPI_COMM_WORLD, ierror)
        end if
      End do
C
      Do for each sending_PE to myid
C
c         Wait until the message is received.
C
        Call MPI_WAIT (ireq(i), istat, ierror)
      End do
```

The equivalent pre-conditioner and solver for an unstructured grid of FEMWATER is an area of further investigation.

The unsaturated flow case is much more difficult. However, a benchmark hard-to-converge unsaturated flow problem was also successfully solved by not completely solving the system of linear equations (100 iterations maximum) at each nonlinear iteration. This technique led to less overall computation time and avoided the challenge of completely solving a linear system with material properties varying several orders of magnitude.. Table 4 gives results for this problem for one time step using the block partition. Note again that the scaled speed-up results are excellent.

2.6 Data Types

FEMWATER runs in double precision mode on a workstation and the SP, and single precision on the T3E.. What allowed the code to run correctly without change on both these systems is first REAL * 8 is used for all the floating point variables, and REAL8 is used as the data type in the MPI calls (see Table 3).

3. Conclusion

It is possible with careful handling of the ghost nodes and border elements to achieve good scaled performance on unstructured-grid flow codes with a minimum of changes to the original legacy program. What pre-conditioners / solvers work best on a given computing platform is still an area of investigation yet to be resolved.

Acknowledgments

This work was conducted by the U.S. Army Waterways Experiment Station, and was funded through the DoD High Performance Computing Modernization Program's CHSSI Project, EQM-3. The training provided by Prof. Zhu, Mississippi State University, and Prof. Mary Wheeler and team, University of Texas, Austin, was very helpful. Permission to publish this paper was given by the Chief of Engineers, U.S. Army Corps of Engineers. The support of SG / Cray Research and the CEWES Major Shared Resource Center are also gratefully acknowledged.

References

1. Lin, H.J., Richards, D.R., Talbot, C.A., Yeh, G.T., Cheng, J.R., Cheng, H.P., and Jones, N.L., *FEMWATER: A Three-Dimensional Finite Element Computer Model for Simulating Density-Dependent Flow and Transport in Variably Saturated Media*, Technical Report CHL-97-12, U.S. Army Engineer Waterways Experiment Station, MS, July 1997.

2. Groundwater Modeling Team, *GMS*, http://chl.wes.army.mil/software/gms/ , U.S. Army Engineer Waterways Experiment Station, MS.

3. Karypis, G., METIS, http://www-users.cs.umn.edu/~karypis/metis/metis.html , University of Minnesota, MN.

Table 4. Unsaturated Flow Results for the T3E

Grid Size	No. of PE's	Nonlinear Iterations	Time (sec)	Scaled Speed-Up (%)
121 X 5 X 81	4	114	777.1	~
121 X 10 X 81	8	114	861.8	90.2
121 X 20 X 81	16	115	945.7	82.2
121 X 40 X 81	32	115	1015.6	76.5
121 X 80 X 81	64	118	1149.3	67.6